Ontology Engineering Group
Laboratorio de Inteligencia Artificial
Dept. Inteligencia Artificial, Facultad de Informática – UPM

OEG Publication

Gómez-Pérez A

Evaluation of Taxonomic Knowledge in Ontologies and Knowledge Bases

Banff Knowledge Acquisition for Knowledge-Based Systems (KAW'99)
Proceedings of the Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. KAW'99 Volumen: II
Editor: Gaines, B.R.; Musen, M.
Editorial: University of Calgary, Alberta, Canadá
16-21 de octubre de 1999.
Banff, Alberta, Canadá.
Pages: 6.1.1 - 6.1.18

Evaluation of Taxonomic Knowledge in Ontologies and Knowledge Bases

Asunción Gómez-Pérez
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn.
Boadilla del Monte, 28660. Madrid, Spain.
asun@delicias.dia.fi.upm.es ,asun@fi.upm.es
Phone: 34-91-3367439
Fax: 34-91-3367412

ABSTRACT

The evaluation of ontologies is an emerging field. At present, there is an absence of a deep core of preliminary ideas and guidelines for evaluating ontologies. This paper presents a brief summary of previous work done on evaluating ontologies and the criteria (consistency, completeness, conciseness, expandability and sensitiveness) used to evaluate and assess ontologies. It also addresses the possible types of errors made when domain knowledge is structured in taxonomies in an ontology and in knowledge bases: circularity errors, exhaustive and non-exhaustive class partition errors, redundancy errors, grammatical errors, semantic errors and incompleteness errors. It also describes the process followed to evaluate the Standard-Units ontology already published at the Ontology Server.

1. INTRODUCTION

During recent years, considerable progress has been made in developing the conceptual bases for building technology that allows knowledge reuse and sharing. Ontologies generally specify conceptualizations with a high degree of formality. However, the process of building ontologies is often anarchistic because each development team usually follows its own set of principles, design criteria and steps in the ontology development process. Until now, few domain-independent methodological approaches (Fernández et al., 1999; Gómez-Pérez 1998; Grüninger and Fox, 1995; Uschold and Grüninger, 1996) have been reported for building ontologies. These methodologies have in common that they start from the identification of the purpose of the ontology, the need for domain knowledge acquisition, and the need for ontology evaluation. However, evaluation is performed differently in each one. Uschold's methodology (Uschold and Grüninger, 1996) includes the evaluation activity but does not state how it is to be carried out. Grüninger and Fox (Grüninger and Fox, 1995) propose to evaluate the ontology by identifying a

set of competency questions, which are the basis for a rigorous characterization of the knowledge that the ontology has to cover. Finally, METHONTOLOGY (Fernández et al., 1999; Gómez-Pérez 1998) proposes that evaluation be carried out throughout the entire lifetime of the ontology development process. It proposes that most of the evaluation be carried out during the conceptualization phase to prevent errors and their propagation in the implementation phase.

However, the Ontological Engineering community is lack of interest in evaluation issues. There exist: a lack of mechanisms to evaluate knowledge sharing technology in general; a lack of specific methods for evaluating existing ontologies and ontologies under construction; a lack of papers describing how very well-known and large ontologies (Cyc Ontologies (Lenat and Guha, 1990), ontologies at the Ontology Server (Farqhuar et al., 1995), SENSUS (Swartout et al., 1997) etc.) have been evaluated before being made public; there is no deep knowledge of how tools for building ontologies perform evaluation; and a lack of application-dependent and end-user methods to judge the usability and utility of an ontology to be used in an application. All of them are obstacles to their use in companies.

Over the years, the Knowledge-Based Systems (KBS) verification and validation community has developed a wide range of methods, techniques and tools for verifying and validating KBS whose KB is formalized as rules. However, ontologies are formalized using concepts organized in taxonomies, instances, relations, functions and axioms in languages like Ontolingua (Gruber, 1993a), CycL (Lenat and Guha, 1990), LOOM (MacGregor, 1991), etc. Therefore, we need specific methods for evaluating them.

Since the evaluation of ontologies is an emerging field and there is an absence of a deep core of preliminary ideas, this paper is organized in two parts. The first part offers a brief review of the work done on evaluating knowledge sharing technology (section 2), ontologies (section 3) and assessing ontologies (section 4). The second part presents the evaluation of taxonomic knowledge in ontologies (section 5) built using primitives from the Frame Ontology (Gruber, 1993a) and the evaluation of the Standard-Units (Gruber and Olsen, 1994) ontology (section 6), both of which are available at the Ontology Server. This paper does not propose to show how to evaluate an ontology as it is built. Its goal is:

1. To present what types of errors should be avoided when building taxonomic knowledge into an ontology under a frame-based approach or, from another viewpoint, the errors to be looked for if an existing ontology is to be reused in another application or ontology.
2. To show what activities should be performed to evaluate an existing ontology before its use by other ontologies and other applications. The Standard-Units ontology was evaluated bearing in mind its future use by the chemical-elements ontology (Fernández et al., 1999).

2. EVALUATION OF KNOWLEDGE SHARING TECHNOLOGY

A study of KBS evaluation ideas served as a precedent for the evaluation of ontologies in order to learn from KBS successes and mistakes (Gómez-Pérez, 1994b). Briefly, the main ideas taken from this study in order to prepare a framework for ontology evaluation are: (1) division into two evaluation types: technical evaluation, carried out by developers, and user evaluation; (2) provision of a set of terms and standard definitions of such terms; (3) definition of a set of criteria to carry out the technical and user evaluation processes; (4) inclusion of evaluation activities in methodologies for building ontologies; (5) construction of tools for evaluating existing

ontologies; and (6) inclusion of evaluation modules in tools used to build ontologies.

With regard to terminology and definitions of terms in the knowledge sharing technology field, the main terms (borrowed from the KBS evaluation field) are (Gómez-Pérez et al., 1995): "evaluation", "verification", "validation" and "assessment".

Evaluation of Knowledge Sharing Technology means to judge the ontologies and Problem Solving Methods (PSMs), their software environments and documentation technically against a reference framework during each phase and between phases of the life cycle. Examples of reference frameworks are the real world, a set of requirements or a set of competency questions. The term evaluation subsumes verification and validation.

Verification of Knowledge Sharing Technology refers to the technical activity that guarantees the correctness of an ontology and PSMs, their associated software environments and documentation with respect to a reference framework during each phase and between phases of the life cycle.

Validation of Knowledge Sharing Technology guarantees that the ontologies and PSMs, their software environments and documentation correspond to the systems that they are supposed to represent.

Assessment of Knowledge Sharing Technology refers to the usability and usefulness of the ontologies and PSMs, their software environments and their documentation when they are reused or shared in applications.

In relation to the criteria for evaluating Knowledge Sharing Technology, a few criteria and examples of evaluation of class definitions, hierarchies and relations and axioms appear in (Gómez-Pérez, 1994a; Gómez-Pérez, 1996). With regard to the tools, the Ontology Server provides a parser for legal KIF sentences. It analyses whether definitions are well formed and generates a report on undefined concepts and intra-ontology dependencies.

3. EVALUATION OF ONTOLOGIES

The evaluation of ontologies (Gómez-Pérez, 1996) refers to the correct building of the content of the ontology, that is, ensuring that its definitions (a definition is written in natural language and in a formal language) correctly implement ontology requirements and competency questions or perform correctly in the real world. The goal is to prove compliance of the world model (if it exists and is known) with the world modeled formally. Ontology evaluation includes:

- Each individual definition and axiom.
- Collection of definitions and axioms that are stated explicitly in the ontology.
- Definitions that are imported from other ontologies.
- Definitions that can be inferred using other definitions and axioms.

The goal of the evaluation process is to determine what the ontology defines correctly, does not define or even defines incorrectly. We also have to look at the scope of the definitions and axioms by figuring out what can be inferred, cannot be inferred or can be inferred incorrectly. To evaluate a given ontology, the following criteria were identified: consistency, completeness, conciseness, expandability and sensitiveness. For examples that show how to deal with these criteria, see (Gómez-Pérez, 1994a; Gómez-Pérez, 1996).

Consistency refers to whether it is possible to obtain contradictory conclusions from valid input definitions. A given definition is consistent if and only if the individual definition is consistent and no contradictory sentences can be inferred using other definitions and axioms.

- A given definition is individually consistent if and only if:
  - the formal definition is metaphysically consistent, that is, if there is no contradiction in the interpretation of the formal definition with respect to the real world. The goal is to prove compliance of the world model (if it exists and is known) with the world modeled formally.
  - the informal definition is metaphysically consistent, that is, if there is no contradiction in the interpretation of the informal definition with respect to the real world.
  - the entire definition is internally consistent, that is, the formal and informal definition have the same meaning.
- A definition is inferentially consistent if it is impossible to obtain contradictory conclusions using the meaning of all the definitions and axioms in the ontology, and the ontologies included by this ontology.

Completeness. Incompleteness is a fundamental problem in ontologies. In fact, we cannot prove either the completeness of an ontology or the completeness of its definitions, but we can prove both the incompleteness of an individual definition, and thus deduce the incompleteness of an ontology, and the incompleteness of an ontology if at least one definition is missing with respect to the established reference framework. So, an ontology is complete if and only if:

- All that is supposed to be in the ontology is explicitly set out in it, or can be inferred.
- Each definition is complete. This is determined by figuring out: (a) what knowledge the definition defines or does not explicitly define about the world; and (b) for all the knowledge that is required but not explicit, check whether it can be inferred using other definitions and axioms. If it can be inferred, the definition is complete. Otherwise, it is incomplete.

In order to provide a mechanism to evaluate completeness, the following activities can be of assistance in finding incomplete definitions.

- Check completeness of the class hierarchy. Errors appear when: the superclasses of a given class are imprecise or over-specified, and when information about subclasses that are subclass partition[1] or about exhaustive subclass partitions[2] is missing.
- Check the completeness of the domains and ranges of the functions and relations. The goal is to figure out whether the domain and range of each argument of each function or relation exactly and precisely delimits the classes that are appropriate for that argument. Errors appear when the domains and ranges are imprecise or over-specified.
- Check the completeness of the classes. The aim is to ascertain whether the class contains as much information as required. Errors appear when: there are properties missing in the definition of a class, when different classes have the same formal definition, etc.

[1] A subclass-partition of a class C is a set of subclasses of C that are mutually disjoint.
[2] A exhaustive subclass partition of a class C is a set of mutually-disjoint classes (a subclass partition) which covers C. Every instance of C is an instance of exactly one of the subclasses in the partition.

Conciseness. An ontology is concise if it does not store any unnecessary or useless definitions, if explicit redundancies do not exist between definitions, and redundancies cannot be inferred using other definitions and axioms.

Expandability refers to the effort required in adding new definitions to an ontology and more knowledge to its definitions, without altering the set of well-defined properties that are already guaranteed.

Sensitiveness relates to how small changes in a definition alter the set of well-defined properties that are already guaranteed.

4. ASSESSMENT OF ONTOLOGIES

Nowadays, it is easy to get information about organizations that have ontologies using the WWW. There are even specific points that gather information about ontologies and have links to other web pages containing more explicit information about such ontologies: The Ontology Page[3] and the yellow pages of ontologies[4] (Arpírez et al., 1998). Additionally, there are ontology servers, like the Ontolingua Server (Farqhuar et al., 1995), Cycorp´s Upper CYC Ontology Server (Lenat and Guha, 1990) or Ontosaurus (Swartout et al., 1997) that collect a huge number of very well-known ontologies.

When developers search for candidate ontologies for their application, they face a complex multi-criteria choice problem. Apart from the dispersion of ontologies over several servers, (a) ontology content formalization differs depending on the server at which it is stored, (b) ontologies on the same server are usually described with different detail levels, (c) there is no common format for presenting relevant information about the ontologies so that users can decide which ontology best suits their purpose, (d) there is no technical document describing how the ontology was evaluated. Choosing an ontology that does not match system needs properly or whose usage is expensive may force future users to stop reusing the ontology and oblige them to formalize the same knowledge again.

So, assessment is focused on judging the understanding, usability, usefulness, abstraction, quality and portability of the definitions from the user point of view, and different kind of users and different kind of applications require different means of assessing an ontology. Some work has been done on identifying a set of features to characterize ontologies from the user point of view (Arpírez et al., 1998; Uschold, 1998). There is also an ontology-based WWW broker, named (ONTO)^{2}Agent(Arpírez et al., 1998), for selecting ontologies that satisfy a given set of constraints.

For example, before reusing definitions from an ontology to build a KBS, the Knowledge Engineer should evaluate questions like: Does the ontology development environment provide methods and tools that help to design the new knowledge base? Does the ontology server include a tutorial or case studies that reduce the cost required to learn and assimilate ontology content? By how much does the ontology reduce the bottleneck in the knowledge acquisition phase? Does the ontology server have translators that transform the ontology content into the target language that I need? How much knowledge is lost in this translation process? Has the ontology been

[3] http://www.medg.lcs.mit.edu/doyle/top
[4] http://delicias.dia.fi.upm.es/OntoAgent/

technically evaluated? How trustworthy is the translation of the definitions into the chosen target language of the KB? And is it possible to integrate the definitions into the KB without making significant modifications to the KB?

5. EVALUATION OF TAXONOMIC KNOWLEDGE IN ONTOLOGIES

Most of the existing ontologies are formalized under a frame-based approach and domain knowledge is structured in taxonomies. This section presents a set of errors that can be made when building taxonomies. This study is based on the experience of evaluating ontologies at the Ontology Server, and should be used to prevent this kind of errors occurring when developing new ontologies. Throughout this section, we will use the semantics of the primitives defined at the frame ontology (Gruber, 1993).

5.1. The Frame Ontology Primitives

The Ontology Server is one of the most commonly used tools for ontology development. Ontology Server ontologies are written in Ontolingua. Ontolingua ontologies were originally written in KIF (Genesereth and Fikes, 1992). A knowledge representation ontology, named the Frame Ontology, was built on KIF. Its goal was to unify the semantics of the most commonly used primitives in the frame paradigm and enable future ontology developers to develop ontologies using a frame-based approach. When building a taxonomy using the frame ontology vocabulary, the following primitives are used:

- *(Subclass-of ?child-class ?parent-class)*: the class ?child-class is a subclass of the class ?parent-class
- *(Superclass-of ?parent-class ?child-class)*: the class ?parent-class is a superclass of the class ?child-class.
- *(Instance-of ?individual ?class):* the instance ?individual is an instance of the class ?class.
- *(Class-Partition ?set-of-classes):* defines a set of disjoint classes. A partition of the class C into the set of classes class_p_1, ..., class_p_n, where class_p_i ≠ class _p_k for every i ≠ k is defined when any instance or subclass of any class class_p_i cannot belong to any other class class-p_k. When this occurs, the classes class_p_1, ..., class_p_n are a partition of class C.
- *(Subclass-Partition ?C ?Class-partition)*: defines the set of disjoint subclasses ?Class-Partition as subclasses of the class ?C. This classification does not necessarily have to be complete, that is, there may be instances of ?C that are not included in any of these subclasses.
- *(Exhaustive-Subclass-Partition ?C ?Class-Partition)*: defines the set of disjoint subclasses ?Class-Partition as subclasses of the class ?C, where the class ?C can be defined as a union of all the classes that make up the partition.

Partitions can define concept classifications in a disjoint and/or complete manner. As exhaustive subclass partitions merely add the completeness constraint to the established subsets, they have been distinguished as: non-exhaustive subclass partition errors and exhaustive subclass partition errors.

5.2. Errors in developing taxonomies

This section presents a set of possible errors that can be made by ontologists when building taxonomic knowledge into an ontology or by Knowledge Engineers when building KBs under a frame-based approach. They are classed under the criteria previously identified in section 3 as: inconsistency errors, incompleteness errors and redundancy errors.

5.2.1. Inconsistency Errors

Circularity errors. They occur when a class is defined as a specialization or generalization of itself. Depending on the number of relations involved, circularity errors can be classed as: circularity errors at distance zero (a class with itself), circularity errors at distance 1 and circularity errors at distance n.

Partition Errors. Partitions can define concept classifications in a disjoint and/or complete manner. As exhaustive subclass partitions merely add the completeness constraint to the established subsets, subclass partition errors and exhaustive subclass partition errors have been distinguished:

- *Subclass partition with common instances.* It occurs when one or several instances belong to more than one subclass of the defined partition. For example, if *dogs* and *cats* form a subclass partition of the set of *mammals*, an error of this type would occur if we define *Pluto* as an instance of both classes. The developer should remove the wrong relation to solve this problem.
- *Subclass partition with common classes* occurs when there is a partition $class\_p_1$,... , $class\_p_n$ defined in a class *class_A* and one or more classes $class\_B_1$, ..., $class\_B_k$ are subclasses of more than one subclass $class\_p_i$ of the partition. For example, if *dogs* and *cats* form a subclass partition of the set of *mammals*, an error of this type would occur if we define the class *Doberman* as a subclass of both classes. The developer should remove the wrong relation to solve the problem.
- *Exhaustive subclass partition with common instances.* It occurs when one or several instances belong to more than one subclass of the defined exhaustive partition. For example, having defined the classes *odd* and *even* as an exhaustive subclass partition of the class *number,* an error of this type appears if the number *four* is an instance of the *odd* and *even* numbers.
- *Exhaustive subclass partition with common classes* occurs when there is a partition $class\_p_1$,... , $class\_p_n$ defined in a class *class_A* and one or more classes $class\_B_1$, ..., $class\_B_k$ are subclasses of more than one subclass $class\_p_i$ of the partition. For example, having defined the classes *odd* and *even* as an exhaustive subclass partition of the class *number,* an error of this type appears if the class *prime* is a subclass of the *odd* and *even* numbers. So, if we define the number *Three* as a prime number, we get the inconsistency since three would be an instance of odd and even numbers.
- *Exhaustive subclass partition with external instances.* These errors occur when having defined an exhaustive subclass partition of the base class (Class_A) into the set of classes class-p_1 ... class-p_n, there are one or more instances of the class_A that do not belong to any

class class_p_i of the exhaustive partition. For example, if the *numbers* classed as *odd* and *even* had been defined as forming an exhaustive subclass partition and the number four were defined as an instance of the class *numbers* (instead of the class *even*), we would have an error of this type.

Semantic Inconsistency Errors. They usually occur because the developer makes an incorrect semantic classification, that is, classes a concept as a subclass of a class of a concept to which it does not really belong; for example, classes the concept *dog* as a subclass of the concept *house*. The same would occur with the instances; for example, if the instance *Pluto* would be an instance-of of the class *house*.

5.2.2. **Incompleteness Errors.** Errors of this type are made when:

Incomplete Concept Classification. Generally, an error of this type is made whenever concepts are classed without accounting for them all, that is, concepts existing in the domain are overlooked. An error of this type occurs if a concept classification *musical instruments* is defined considering only the classes formed by *string instruments* and *wind instruments* and overlooking, for example, the *percussion instruments*.

Partition Errors. They could appear when the definition of the partition between a set of classes is omitted. We have identified two types of errors:

- *Subclass partition omission.* The developer identifies the set of subclasses of a given class, but omits that the subclasses are disjoint. An example would be to define *dogs* and *cats* as a subclass of *mammals* and to omit that *dogs* and *cats* form a subclass partition (though not complete) of the set of *mammals*.
- *Exhaustive subclass partition omission.* The developer defines a partition of a class and omits the completeness constraint to the established subsets. Examples would be to define *odd* and *even* as a subclass of *numbers* or to define *odd* and *even* as a subclass partition of *numbers*. In both cases, it is omitted that the *numbers* classed as *odd* and *even* form an exhaustive subclass partition (that is, complete).

5.2.3 Redundancy Errors. Redundancy is a type of error that occurs when redefining expressions that were already explicitly defined or that can be inferred using other definitions.

Gramatical Redundacy Errors. These errors occur in taxonomies when there is more than one explicit definition of any of the hierarchical relations.

- *Redundancies of subclass-of relations* occur between classes when subclass-of relations are repeated. We can distinguish direct and indirect repetition. It exists a *Direct repetition*, when two or more subclass of relations between the same source and target classes are defined, that is, including the subclass of relation between the classes *dog* and *mammals* twice. It exists a *Indirect repetition,* for example, if we define the class *dog* as a subclass of *pet,* and *pet* as a subclass of *animal,* when *dog* is also defined as a subclass of *animal.*
- *Redundancies of instance-of relations.* As in the above case, there are two possibilities. It exists a *direct repetition* if two *instance-of* relations between the same instance and class are defined. It exists an indirect repetition, for example, if we define the instance *Clyde* as an

instance of *real elephant* and *real elephant* as a subclass of the class *elephant*. The definition of an instance of relation between *Clyde* and *elephant* would lead to a redundancy in the taxonomy.

Identical formal definition of some classes. It occurs when there are two or more classes in the ontology with the same formal definition, that is, the only difference between the subclasses is the name. This error type could be also classified as an example of classes with incomplete knowledge. The developer could solve this problem by adding what distinguishes the classes of the partition or, otherwise, realize that it does not make sense to have classes with identical formal definitions in the partition and delete one of them.

Identical formal definition of some instances. It occurs when there are two or more instances in the ontology with the same formal definition, that is, the only difference between them is the name.

6. EVALUATION OF THE ONTOLOGY STANDARD-UNITS

6.1 Scope

This section presents how the approach outlined in the previous sections is applied to the evaluation of the Standard-Unit (SU) ontology (Gruber and Olsen, 1994). Gruber's five design criteria[5] (Gruber, 93b) and the Ontological Distinction Principle[6] (Borgo et al., 96) that have proved useful in the development of ontologies were also applied in the evaluation of the SU ontology.

The SU ontologies was selected for evaluation due to the following reasons. First, the SU ontology is a simple ontology that only provides classes and instances. Second, we came to revise the SU ontology because it was included in chemical-elements ontology (Fernández et al., 1999). To evaluate the Standard-Units (SU) ontology, we took its Ontolingua implementation. Then, we obtained a possible conceptual model and we analyzed it. After this, we corrected and restructured the initial conceptual model and we created a new conceptual model, which has been re-implemented. The new implementation was also evaluated against the established reference framework.

6.2 Background knowledge

The SU ontology defines a series of SI units of measurement and other commonly used units that do not belong to the SI units. It includes the Standard-Dimensions ontology, which defines a

[5] *Clarity and Objectivity*, which means that the ontology should provide the meaning of defined terms by providing objective definitions and also natural language documentation; *Completeness*, which means that a definition expressed by a necessary and sufficient condition is preferred over a partial definition (defined only by a necessary or sufficient condition); *Coherence*, to permit inferences that are consistent with the definitions; *Maximize monotonic extendibility*, which means that new general or specialized terms should be included in the ontology in a such way as does not require the revision of existing definitions; and *Minimal ontological commitments*, which means making as few claims as possible about the world being modeled, giving the parties committed to the ontology freedom to specialize and instantiate the ontology as required.

[6] Classes corresponding to different identity criteria must be disjoint.

series of physical dimensions (i.e., mass, time, length, temperature and electrical current) for different quantities. It also includes other dimensions, derived from the above five, including pressure, volume, etc. Depending on the system of units used, the physical quantities defined at the Standard-Dimensions ontology can be expressed in different units using the vocabulary of the SU ontology; for example, length can be expressed in meters, miles, inches, etc. Both the SU and the Standard-Dimensions ontologies include the Physical-Quantities ontology, which defines the basic vocabulary for describing physical quantities in a general form, making explicit the relationship between quantities of various orders, units of measure and physical dimensions. A quantity is a hypothetically measurable amount of something. For example, the term meter, defined in the SU ontology, is an instance of the class Unit-Of-Measure defined in the Physical-Quantities ontology.

The SU ontology was analyzed on the basis of its Ontolingua implementation. Figure 1 shows a preliminary conceptual model that possibly originated such implementation. It is important to note that this ontology contains neither relations, functions nor axioms. The hierarchy illustrates that there are two classes: Unit-Of-Measure and System-of-Units, both defined in the Physical-Quantities ontology. In this manner, a series of units of measure which are instances of the class Unit-Of-Measure are defined in SU, as well as a class, Si-Unit, which groups all the SI units. In the SU ontology, all the units have a property that indicates the dimension of the aforesaid unit. These dimensions are defined in the Standard-Dimensions ontology. In this ontology, there is a class, called Physical-Dimension, which is also defined in the Physical-Quantities ontology, of which all the dimensions defined are instances in the Standard-Dimensions ontology (figure 2).

We came to revise the SU ontology because it was included in Chemical-Elements. We needed to check that the units of measurement of certain attributes in Chemical-Elements befitted the knowledge and usual practice of experts. The experts had drawn up the inspection document setting out the properties to be checked, and we transformed them into the vocabulary of the ontology. For example, check that the Semidisintegration-Period of the concept Elements is filled in with a value type Time-Quantity and its unit of measurement is Year (see figure 3).

6.3 Evaluation of the Standard Unit Ontology

This ontology was evaluated in two steps -Analysis and Synthesis. It is in the analysis phase when the ontology is evaluated and assessed. The synthesis phase seeks to correct the ontology and document any changes made. Two ontologists and two domain experts were involved in analyzing and synthesising the SU ontology.

6.3.1. Analysis. Taking into account figure 1 and the SU Ontolingua code (version available at the Ontology Server as of December 1997), the SU ontology contains 61 instances and one class. As we said before, all the units are defined as instances of the class Unit-of-Measure and the SI units are defined also as an instance of the class SI-Units. Therefore, this ontology provides a poor taxonomy of organizing ontological knowledge. This situation causes at least two problems. First, the instances cannot be classified by similar characteristics. Second, part of the inference power allowing some concepts to inherit properties from more general concepts in a properly diversified hierarchy is lost. It would be advisable to build branched taxonomies to take advantage of the above-mentioned benefits.

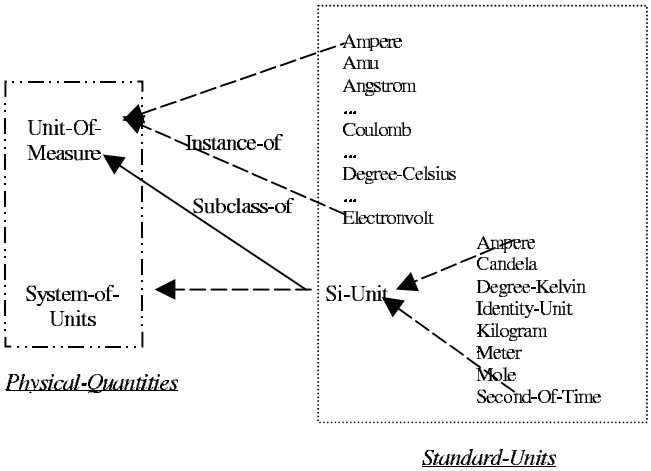

Figure 1. Preliminary hierarchy of the Standard-Units ontology.

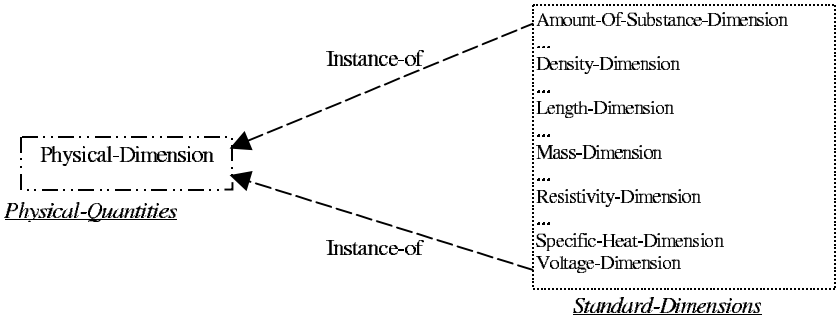

Figure 2. One of the hierarchies of the Standard-Dimensions ontology.

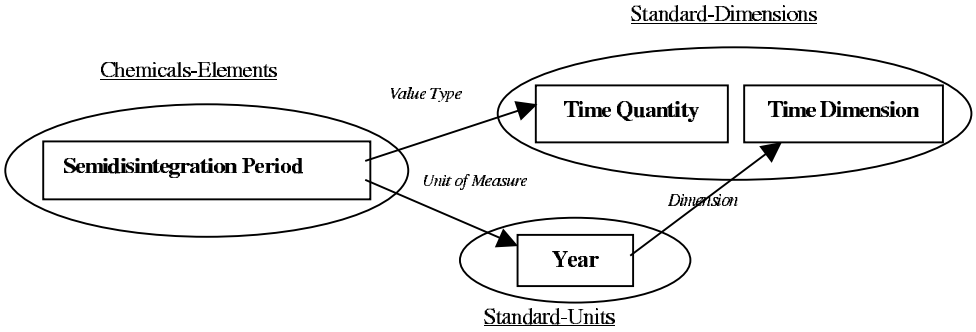

Figure 3. Relation between the Standard-Units and Standard-Dimensions ontologies.

A) Analysis of the taxonomy. Following the approach outlined in section 5.2, we can say the following about the hierarchical organization of the ontological knowledge.

- *Analysis of inconsistencies.* The ontology is consistent. There is not circularity errors at distance zero, there is not partition errors since it is impossible to define a partition using a unique class, and there is not semantic inconsistency.
- *Analysis of incompleteness.* The ontology is incomplete since there is no taxonomic organization classifying the units of measurement according to some criterion that divides the general concepts into other more specific concepts all the way down to instances. By contrast, there is a single class to which all the instances are subordinated.
- *Analysis of conciseness.* The ontology is concise since hierarchical relations between each instance and the class is not repeated. This ontology does not provide identical formal definitions of some instances.

B) Analysis of the Ontology. Taking into account the criteria presented in (Gruber, 1993b) and the evaluation examples presented in (Gómez-Pérez, 1996), the inspection of the whole ontology code has shown other problems that are obstacles to understand and extend the ontology. They are:

1. Understandability and extendibility problems. Definitions that should be made in the same manner, as they refer to similar concepts, are made differently in the Ontolingua code. For example, the SI base unit called "Ampere" was defined:

```
(Define-Frame Ampere
        : Own-Slots
        (( Documentation "Si electrical current unit.")
         (Instance-Of Unit-Of-Measure)
         (Quantity.Dimension Electrical-Current-Dimension))
        : Axioms
        ((= (Quantity.Dimesion Ampere) Electrical-Current-Dimension)))
```

However, the definition of the SI base unit named "Meter" is:

```
(Define-Instance Meter (Unit-Of-Measure)
        "SI length unit. No conversion is given because this is a standard."
        : Axiom-Def
        (And (= (Quantity.Dimesion Meter) Length-Dimension)
           (Si-Unit Meter)))
```

It would be advisable for definitions that are children of the same parent (SI-Unit) to be defined according to the same templates to improve ontology understanding and to ease the inclusion of new definitions. So, the lack of patterns for doing similar definitions makes difficult the assessment of the ontology by the end-user and also the expandability of the ontology.

2. The choice of names for the different instances does not comply with a fixed standard. For example, the different multiples and divisors of Ampere are called: Milli-Amp, Nano-Ampere and Pico-Ampere. To ease ontology understanding and improve its *clarity*, the same naming conventions should be used to name related terms. Therefore, the above-mentioned names

should be standardized and denoted as follows: Milli-Ampere, Nano-Ampere and Pico-Ampere.

3. The multiples of the base units do not appear to have been chosen systematically. For instance, Kilo-ohm and Milli-meter are omitted. When restructuring the SU ontology, our framework was the set of units of interest for the Chemical-Elements ontology. As Kilo-ohm and Milli-meter will not be used in our framework, we can say that the SU ontology is complete in this framework of reference.

4. Incompleteness in formal definitions. The ontology includes factors of conversion between different units of the same dimension. However, this conversion is not always made from one particular unit to the unit that is considered the base unit at the SI. For example, taking the base unit of time "second", each definition of its multiples (minutes, hours, etc.) and its submultiples (millisecond, microsecond, etc.) should contain the appropriate factor of conversion to seconds. However, definitions appear in the SU ontology with factors of conversion as follows:

```
(Define-Frame Day
        : Own-Slots
        ( (Documentation "one day, i.e., 24 hours")
         (Instance-Of Unit-Of-Measure)
         (Quantity.Dimension Time-Dimension) )
        : Axioms
        ( (= Day (* 24 Hours)) (= Year (* 365 Day) ) ) )
```

In this definition, the factors of conversion of the unit "Day" are established in relation to non-base units (hours and years), but not with the base unit (seconds). The following factor of conversion should be added to the formal definition:

```
( (= Day (* 86400 Second-Of-Time)))
```

The conversion should always be made to the base unit to improve the clarity of the ontology. Other commonly used factors of conversion between units can also be added, but the conversion to the base unit should never be missing.

5. Incompleteness in the NL documentation. Some definitions have quite a poor informal language description, which provides the user with no information. This is the case of the natural language definition of Meter, which states: "SI length unit. No conversion is given because this is a standard." An extreme example is Kilometer, for which no informal definition is given at all. A natural language definition should be included whenever possible to give a better understanding of the more formal definition made later. In the example, " A Meter is 1650763.73 wave lengths in vacuo of the unperturbed transition $2p_{10}$ - $5d_5$ in ^{86}Kr."

6. Neither was the vocabulary of the Standard-Dimensions ontology used in a standardized manner. Thus, for example, the dimension *Megapascal* is defined as a "Pressure-Dimension"

```
( Quantity.Dimension Megapascal Pressure-Dimension)
```

Whereas the dimension *Pascal* is said to be:

```
( = (Quantity.Dimension Pascal ) (* Force-Dimension (Expt Length-Dimension -2)))
```

As the "Pressure-Dimension" definition exists in the Standard-Dimensions ontology, which is used by SU, it would be more rational and clearer to define all the units of pressure using this dimension, instead of using its equivalent in units of length and force. Therefore, the dimension Pascal should be defined as follows:

(Quantity.Dimension Pascal Pressure-Dimension)

7. In the SU ontology, the number Pi (π) is defined as an instance of the real numbers because a factor of conversion between angles and radians appears in the definition of "Angular-Degree".

(Define-Instance **Angular-Degree** (Unit-Of-Measure)

"Angular measurement unit."

:= (* Radian (/ The-Number-Pi 180))

:Axiom-Def (= (Quantity.Dimension Angular-Degree) Identity-Dimension))

As this is an ontology of units of measure, definitions that have nothing to do with the above units must not be included. This problem could be solved in two ways: one possible solution would be to delete the definition of the number π in the factor of conversion and enter the real number 3.1415926535897936. However, a better solution is to include the real number π in the KIF-Numbers ontology, which could be included in the SU ontology and thus this definition could be used.

6.3.2 Synthesis. After analyzing the SU Ontolingua code and obtaining a possible underlying conceptual model of the ontology and after considering the problems explained above, we modified its conceptual model following the next criteria:

Standardize naming conventions. We gave standard names to the new classes and instances. The names of the classes in the SU ontology were chosen taking into account the type of units represented and the names of the dimensions found in the Standard-Dimensions ontology.

Specialization of concepts. The goal was to identify general concepts that are specialized into more specific and disjoint concepts down to domain instances. The criterion used for specialization was to group units according to the base unit. Therefore, we can say that the restructured ontology complies with the Ontological Distinction Principle (Borgo et al., 1996) which states that classes should be disjoint. We also have maximized the monotonic extendibility (Gruber, 1993a) of the SU ontology because the inclusion of classes and instances does not require the revision of existing definitions.

Branched taxonomies. Whenever possible, the hierarchy must be sufficiently branched by similar characteristics to increase the power provided by inheritance mechanisms between classes and instances. Figure 4 illustrates the new hierarchy.

Inclusion of new properties and changes to existing properties. In this ontology, the property Abbreviation was added to each unit defined for the purpose of extending the use of this international standard for this attribute, ruling out widespread, though not absolutely correct, uses.

Minimize the semantic distance between sibling concepts (Arpírez et al., 98). Similar concepts are usually grouped and represented as subclasses of one class and should be defined using the same set of primitives, whereas concepts which are less similar are presented further apart in the hierarchy. All the terms in the restructured ontology have been defined using the same pattern in order to give a clearer understanding of the ontology. The SI units were all define following the same template. As an example:

```
(Define-Okbc-Frame Kilogram
        :Direct-Types
        (Si-Unit Mass-Unit)
        :Own-Slot-Specs
        ((Documentation "SI mass unit. Its is equal to the mass of the international prototype
of the kilogram.")
        (Quantity.Dimension Mass-Dimension)
        (Abreviation "kg")))
```

For the purpose of assuring information about the evolution of the SU ontology, a rigorous change control (Gómez-Pérez et al., 99) has been performed. The goal is to have all the changes documented and to facilitate future modifications and changes to the ontology. Change control is what allows ontology developers to control the evolution of the ontology, ruling out any errors caused by undesired effects in the final implementation, such as inconsistency and incompleteness. Change control also helps end users to determine which version of the ontology they require for their system or for the new ontology they are to develop.

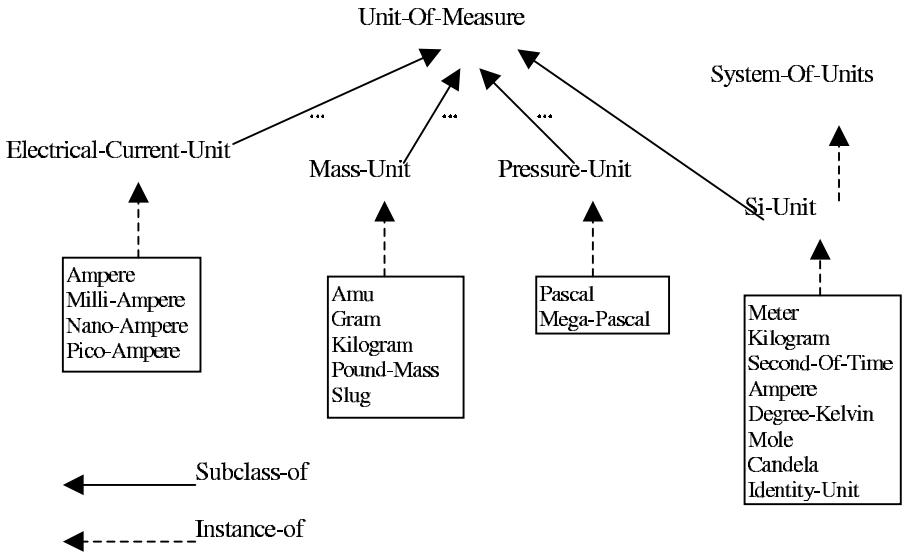

Figure 4. Taxonomy for the Modified Conceptual Model.

6.4 Implementation and Re-Evaluation

On the basis of these considerations, the SU ontology was re-implemented in Ontolingua using the Ontology Server editor. The same people evaluated the re-implemented ontology, and an external person that did not participated on the two previous phases compared them using the previous evaluation framework and common sense knowledge. The conclusions were:

- The ontology is syntactically correct, as it successfully passes the Ontology Server Analyze tests.
- The ontology is complete for the use to which it is to be put in the chemical ontology.
- The ontology is consistent, as the knowledge formalized has been checked against the above-mentioned sources of knowledge.
- The ontology is concise, as there is no redundant knowledge.
- The new implementation is more structured and understandable than the old one.

For the purpose of assuring information about the evolution of the SU ontology, a rigorous change control (Gómez-Pérez et al., 99) has been performed. The goal is to have all the changes documented and to facilitate future modifications and changes to the ontology. Change control is what allows ontology developers to control the evolution of the ontology, ruling out any errors caused by undesired effects in the final implementation, such as inconsistency and incompleteness. Change control also helps end users to determine which version of the ontology they require for their system or for the new ontology they are to develop.

The team took 108 hours to evaluate, re-implement, re-evaluate and to perform the change control documents.

7. CONCLUSIONS

Although there are a host of papers on knowledge-based systems evaluation, the ontology evaluation field is only just emerging. In this paper, we have presented the conceptual bases of ontology evaluation from a technical point of view and an example of how evaluation of taxonomies should be performed. It includes:

- A brief summary of previous work done on ontology evaluation and the criteria (consistency, completeness, conciseness, expandability and sensitiveness) used to evaluate and assess ontologies.
- Identification of a set of errors that can be made when domain knowledge is structured in taxonomies. Errors are classed in the following categories: circularity errors, partition errors (including exhaustive subclass partitions and non-exhaustive subclass partitions), redundancy errors, and semantic errors.
- It also describes the SU ontology evaluation process following the approach presented at this paper and bearing in mind the future use of this ontology by a Chemical-elements ontology.

Future work will include primarily:

- Addressing in more depth the theoretical foundations of ontology evaluation, that could include the evaluation of other components.

- Introducing activities related to ontology evaluation into ontology development methodologies. The purpose of these activities will be to raise ontologist awareness of the fact that evaluation should be performed throughout the entire ontology life cycle in order to detect errors at the earliest possible time and should not be left until the end when the ontology has been implemented.
- Evaluation of very well-known and large ontologies like Cyc, SENSUS, etc.
- Develop independent tools for evaluating ontologies built under a given knowledge representation paradigm.
- Develop evaluation tools to be integrated in the environments used to build ontologies.
- Create application-dependent and end-user methods to judge the adequacy of an ontology for a given application.

ACKNOWLEDGEMENTS.

First part (sections 2 to 5) of this research work has been performed at the Knowledge Systems Laboratory in Stanford University. It has been sponsored by grant number PF94-9921929 of the Ministerio de Educación y Ciencia in Spain. Second part of the paper (section 6) has been supported by the project TIC 96-1226-E funded by the Comisión Interministerial de Ciencia y Tecnología (CICYT) in Spain. I would like to thanks to the following persons: Almudena Galán and Rosario García, for their knowledge of chemistry and the environment; Maria Dolores Rojas for the help on the analysis and sinthesis of the standard units ontology, and Mariano Fernández, for the evaluation of the new version of the standard-units ontology. Thansk to the anonymous reviewer and Mariano Fernández for their comments.

REFERENCES

Arpirez, J.; Gómez-Pérez, A.; Lozano, A.; Pinto, S. (1998). (ONTO)^{2}Agent: An ontology-based WWW broker to select ontologies. In Workshop on Applications of Ontologies and PSMs-Brighton. England. August pp. 16-24.

Borgo, S.; Guarino, N.; Masolo. (1996). C. Stratified Ontologies: the case of physical objects. In proceedings of the Workshop on Ontological Engineering. Held in conjunction with ECAI96. Pages 5-15. Budapest.

Farquhar A., Fikes R., Rice J. (1996). The Ontolingua Server: A Tool for Collaborative Ontology Construction, Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Alberta, Canada, PP. 44.1-44.19.

Fernández, M.; Gómez-Pérez, A.; Pazos, J.; Pazos. A. (1999). Building a Chemical Ontology using methontology and the ontology desing environment. IEEE Intelligent Systems and their applications. !4 (1):37-45.

Genesereth M., Fikes R.. (1992). Knowledge Interchange Format, Technical Report, Computer Science Department, Stanford University, Logic-92-1.

Gómez-Pérez, A.; Rojas-Amaya, M.D. (1999). Ontological Reengineering for Reuse. Knowledge Acquisition Modeling and Management. 11th European Workshop, EKAW'99.

Dagstuhl Castle, Germany, May 26-29, Pags 139-156.

Gómez-Pérez A. (1998). Knowledge Sharing and Reuse, The Handbook of Applied Expert Systems, Edited by J. Liebowitz, CRC Press.

Gómez-Pérez, A. (1996). A Framework to Verify Knowledge Sharing Technology. Expert Systems with Application. Vol. 11, N. 4. PP: 519-529.

Gómez-Pérez, A.; Juristo, N.; Pazos, J. (1995) Evaluation and Assessment of the Knowledge Sharing Technology. Towards Very Large Knowledge Bases. N.J.I. Mars, Ed. IOS Press, 1995.

Gómez-Pérez, A. (1994a).Some ideas and Examples to Evaluate Ontologies. Technical Report KSL-94-65. Knowledge System Laboratory. Stanford University. Also in Proceedings of the 11th Conference on Artificial Intelligence for Applications. CAIA94.

Gómez-Pérez, A. (1994b). From Knowledge Based Systems to Knowledge Sharing Technology: Evaluation and Assessment. Technical Report. KSL-94-73. Knowledge Systems Laboratory. Stanford University. December

Gruber, T. and Olsen, R. (1994). An Ontology for Engineering Mathematics, Technical Report KSL-94-18, Knowledge Systems Laboratory, Stanford University, CA.

Gruber, T. (1993a). A translation Approach to portable ontology specification. Knowledge Acquisition. 5: 199-220.

Gruber T. (1993b). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human Computer Studies. 43:907-928.

Gruninger M., Fox M. (1995). Methodology for the Design and Evaluation of Ontologies, Proceedings of IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing.

Lenat D.B., Guha, R V. (1990). Building Large Knowledge-based systems. Representation and Inference in the Cyc project. Addison-Wesley, Reading, Massachusetts.

MacGregor, R. (1991). Inside the LOOM classifier. SIGART bulletin, 2(3):70-76. June.

B. Swartout, R. Patil, K. Knight and T. Russ. (1997). Towards Distributed Use of Large-Scale Ontologies. Spring Symposium Series on Ontological Engineering. Stanford University, CA. Pages: 138—148.

Uschold M., Grüninger M. (1996). ONTOLOGIES: Principles, Methods and 16Applications, Knowledge Engineering Review, Vol. 11, N. 2, June.

Uschold, M. (1998). Where are Killer Apps? . Workshop on Applications of Ontologies and Problem Solving Methods. ECAI98. Brighton.